AFRL-IF-RS-TR-2000-88
Final Technical Report
June 2000

MULTIPLE PERSPECTIVE INTERACTIVE VIDEO SURVEILLANCE AND MONITORING SYSTEM

Ramesh Jain

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

**AIR FORCE RESEARCH LABORATORY
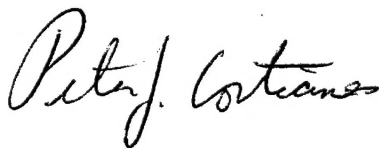INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK**

DTIC QUALITY INSPECTED 4

20000724 025

This report has been reviewed by the Air Force Research Laboratory, Information Directorate, Public Affairs Office (IFOIPA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

AFRL-IF-RS-TR-2000-88 has been reviewed and is approved for publication.

APPROVED: *Peter J. Costianes*

PETER J. COSTIANES
Project Engineer

FOR THE DIRECTOR: *John V. McNamara*

JOHN V. MCNAMARA, Technical Advisor
Information & Intelligence Exploitation Division
Information Directorate

MULTIPLE PERSPECTIVE INTERACTIVE VIDEO SURVEILLANCE AND MONITORING SYSTEM

Ramesh Jain

REPORT DOCUMENTATION PAGE

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	JUNE 2000	Final Jan 97 – Aug 99

4. TITLE AND SUBTITLE
MULTIPLE PERSPECTIVE INTERACTIVE VIDEO SURVEILLANCE AND MONITORING SYSTEM

5. FUNDING NUMBERS
C - F30602-97-1-0077
PE - 62301E
PR - E651
TA - 00
WU - 01

6. AUTHOR(S)
Ramesh Jain

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Department of Electrical and Computer Engineering
University of California, San Diego
La Jolla CA 92093-0934

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Defense Advanced Research Projects Agency   Air Force Research Laboratory/IFED
3701 N. Fairfax Drive   32 Brooks Road
Arlington VA 22203-1714   Rome NY 13441-4114

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

AFRL-IF-RS-TR-2000-88

11. SUPPLEMENTARY NOTES
Air Force Research Laboratory Project Engineer: Peter J. Costiannes/IFED/(315) 330-4030
Defense Advanced Research Projects Agency Program Manager: George Lukes/ISO/(703) 696-2364

12a. DISTRIBUTION AVAILABILITY STATEMENT
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)
This report describes a technique for gathering information from a multitude of sensors in a given environment, and integrating it into a coherent and comprehensive Environment Model (EM). The focus of this effort is on the use of Multiple Perspective Interactive Video technology for intelligent video monitoring of environments. The technology developed was applied to two scenarios: the surveillance of a room used as a library and the surveillance of a parking lot. An overall architecture was developed, one for each scenario. For the first scenario (library), four cameras were used to monitor the activity in the room. The system worked on two PCs and monitored activity like "stealing an object," "loitering near a bookshelf," and "jumping on a table." The second scenario (parking lot), included three cameras. The system monitored activity like pedestrians going near several cars, cars entering the parking lot at excessive speed, and cars remaining in the same spot for more than a specified amount of time.

14. SUBJECT TERMS
Video Surveillance, Computer Vision

15. NUMBER OF PAGES
32

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

Table of Contents

List of Figures

List of Tables

DARPA Video Surveillance and Monitoring, Final Report

Ramesh Jain

Introduction

MPI-Video (Multiple Perspective Interactive Video) is a technique for gathering information from a multitude of sensors in a given environment, and integrating it into a coherent and comprehensive *Environment Model* (EM). MPI-Video was developed at the Visual Computing Laboratory at UCSD as a tool for integrating data from different cameras, and is being pursued both as a research topic and as a commercial opportunity.

MPI-Video addresses two classes of research issues, both interesting and challenging. First, there is the problem of building the EM. This includes traditional problems of sensor fusion, image and video understanding, as well as new challenges integrating data from sensors as diverse as a video camera and a microphone or a GPS receiver.

Second, we have the problem of making effective use of the information provided by the EM. The EM is continuously gathering and integrating information from the environment. There are two types of use that we can do of this information: *real-time* and *historical*. Real-time access allows us to develop agents that inhabit a certain area and to shift the burden of cognition from the agents to the environment itself. As a typical example, consider a robot placed in a certain setting with a goal that requires understanding its surroundings. Rather than building a cognizant robot tailored to the specific setting it is possible to build a robot with a very general goal satisfaction engine that doesn't take into account the problems of navigation (obstacle avoidance, following prescribed routes, and so on). The robot will be in constant contact with the environment and it will repeatedly ask the EM for the suitable move needed to satisfy a very short term plan ("I have to reach position x, y; in which direction should I move in order to avoid undesirable situations?").

Historical access entails accessing information relative to past events. In this case we face the challenges of developing a spatio-temporal database in which the data can be stored. These challenges include designing an appropriate data representation, designing a suitable spatio-temporal query language, designing appropriate indexing structures, and so on. Applications of historical access include the statistical analysis of the usage patterns of a given physical facility, the off-line interactive view of sporting events, and many others.

The focus of this paper is on the use of the MPI-Video technology for intelligent video monitoring of environments. We assume that a number of sensors (typically cameras) have been deployed in a certain area, and that one or more users are interested in monitoring certain behaviors in this environment. This application belongs to both application classes outlined above. On one hand, we are interested in setting real-time alarms if, for instance, somebody is trying to steal something or to enter a restricted area. On the other hand, we want to access historical data both for monitoring purposes (which area of this office has, on average, the highest concentration of people?) and to determine "regular" patterns of behavior that can be used to identify---and set alarms on---irregular behaviors.

We are applying the technology to two scenarios: the surveillance of a room used as a library in our lab (quite aptly called the "Orwell Room"), and the surveillance of a parking lot that we are developing for the VSAM (Video Surveillance and Monitoring) project. The scenarios were selected with certain criteria in mind. First we accept a somewhat simplified situation, but not a toy example: the applications should be realistic. Second, these scenarios poses some meaningful challenges to vision-based systems. For instance, the system has to work reliably under very different illumination conditions and the subjects are non-cooperative (therefore it is not possible to identify them with an active badge or similar wearable devices). Finally, the scenarios present some simplifying circumstances, namely there are relatively few objects, so that tracking the individual objects is feasible (as opposed, for instance, to an airport situation).

1

MPI-VSAM Architecture

The goal of MPI-VSAM is to gather information about the environment, incorporate the information in an EM, and allow users to query the model. The system is divided in two parts: the *acquisition subsystem* and the *query subsystem*. The two systems are joined conceptually by the EM. Roughly, the acquisition subsystem provides the EM with data from the environment that the EM will integrate into a coherent picture of *objects* and *events*. The query subsystem gives the user the tools to access the information in the EM both in real-time and historically (see Figure 1).
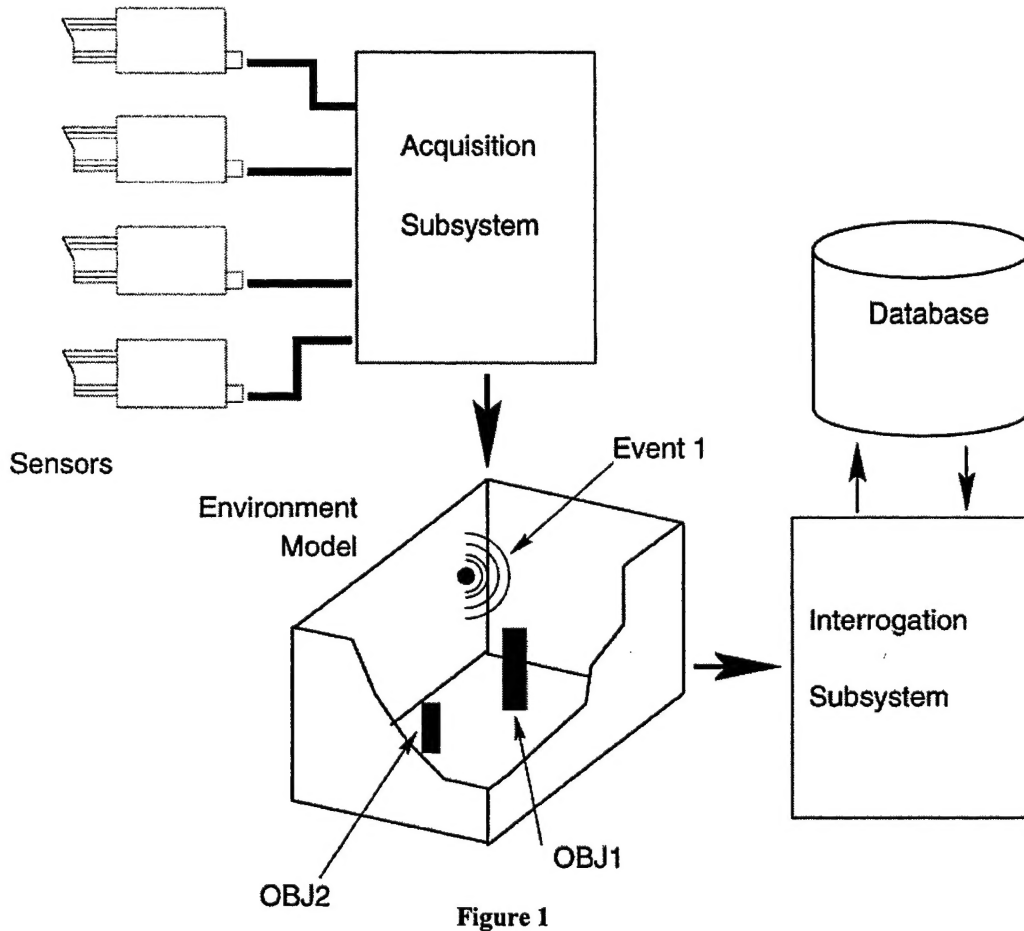
Figure 1

The actual organization of the system is slightly different from this conceptualization. The major difference is that we consider the EM as part of the acquisition subsystem. As we will discuss in the next section, the acquisition subsystem already contains in its components all the information necessary to the EM. All we need to do is to coordinate this information and place in its proper spatio-temporal location. Consider for instance a component whose task is to segment the image from one camera and extract some color information about the objects in the scene. (Figure 2).

The color detection subsystem has a partial view of the object: the position of the object in the 3D space is not known, and other information like the velocity of the object are unknown. This information, however, is built in the EM based on data from all the sensor, and integrated in a *record* containing an object id and all the available information about the object. The object id can be fed back to the color detection subsystem which now is capable of integrating the information (via the object id) with all the other information contained in the EM.

Image Plane

Camera

Color Histogram

Image Plane

Color histogram
Association

Data from
all the
cameras

Obj1
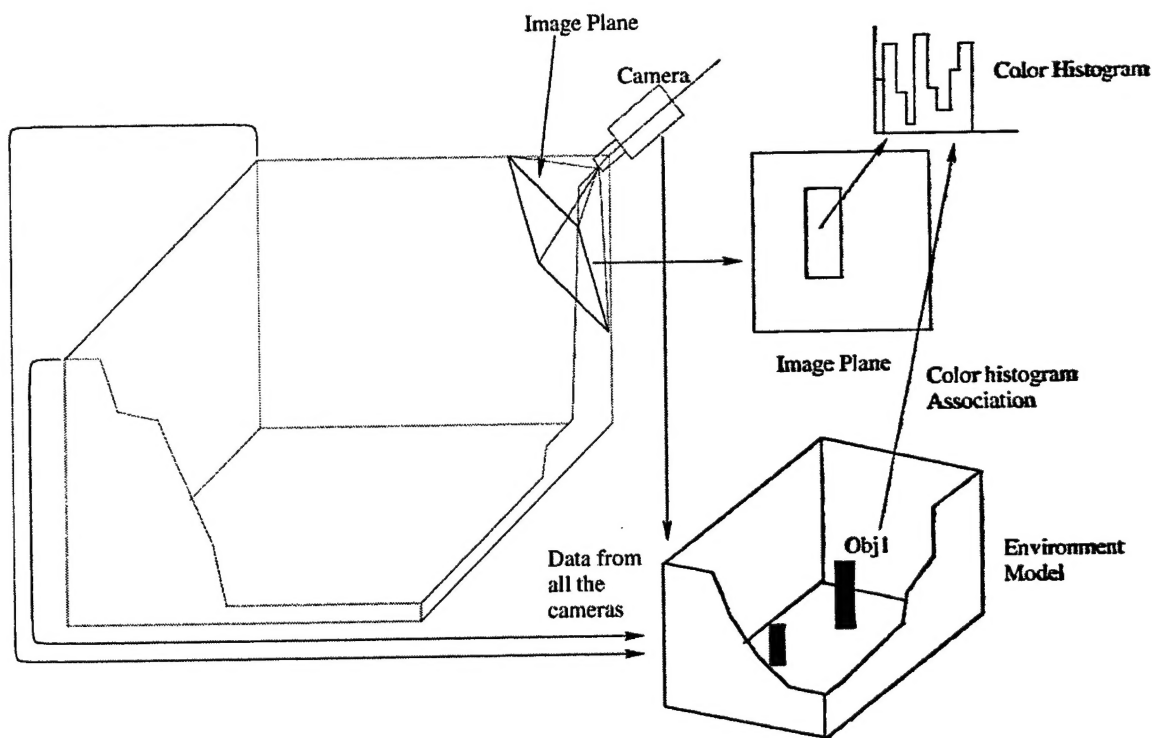
Environment
Model

Figure 2

Acquisition Subsystem

The acquisition subsystem collects data from the sensors placed in the environment, and integrates the data into a coherent picture of the objects and events taking place. This activity is composed of the following operations:

Detection. Events and objects must be detected in the individual data streams. If we have cameras, this activity entails segmenting the objects and detecting interesting low-level events like, for instance, a flash of light (complex events like the encounter of two objects are detected at the EM level, where enough information is available). Detection at this level is in the sensor space. In the case of a camera, this is the image space.

Correspondence. Data from different sensors must be associated. For instance, it is necessary to determine whether two visual objects identified in the image plane of two different cameras represent two views of the same physical object or not.

Localization. The data obtained from the different sensors are used to locate an object in space and time.

Back-projection. Since the objects have been placed in the 3D space and labeled, it is possible to go back to the individual streams and assign additional features to the object. For instance, if we extract color information as part of the segmentation process, it is possible to associate that information to the appropriate object in the EM.

3

These operations are not independent. For instance, finding correspondences between segments in different images is necessary in order to localize objects into the three dimensional space. At the same time, localization helps finding correspondences. Localization and correspondence can be used to extract information (say, a color histogram) from a single object but, at the same time, color characteristics can be used to resolve ambiguities in the correspondences.

The VSAM acquisition subsystem is represented schematically in Figure 3.
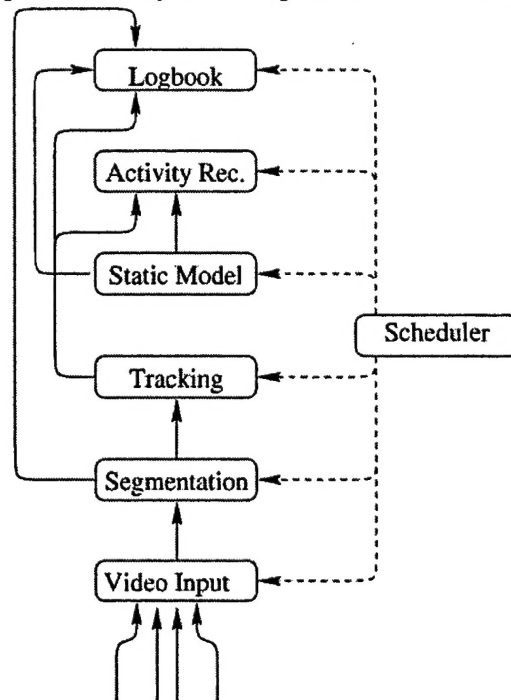
Figure 3

The system is organized in a series of intercommunicating *layers*. Each layer operates in the video stream or on the results of other layers. The layers operate synchronously under the control of a scheduler. Time is discretized naturally according to the granularity of the sensor being served. In the case of video, a time step corresponds to the processing of a frame, in the case of audio to the processing of an audio sample, and so on. The general operation of the layers is modeled after that of the discrete Kalman filter.

Each layer defines two methods: *estimate* (predict the state of the layer at the next time step) and *update* (refine the predicted state using the new data), which are called at every step. Each layer can receive the input from every other layer in the system, and must produce one output for each time step. Data about the environment is collected in a *static model*, a small database whose only data type is the *region*. The static environment is modeled as a collection of *regions of interest*. It provides primitives for defining regions and *constraints* on regions. In the parking lot scenario, for instance, the driveway can have the constraint that no car can stop there for more than a few seconds, while a parking spot might have the constraint that no car can park there for more than two hours. The position, location and other characteristics of every object at every instant in time are collected in a Log-book, which is also a short-time database.

Tracking

Extracting the 3D position and velocity of objects in a scene is of interest in many applications. We have developed the tracking algorithm for use in the surveillance system. 3D

locations and velocities of objects are needed for recognition of activities in the environment such as removing an object from its location, loitering, or speeding through a parking lot.

Most of the methods for tracking people involve estimating positions using models of objects in 2D [i,ii] or 3D [iii,iv,v]. However, we are not aware of real-time algorithms that recover 3D positions of objects from multiple cameras. One can always place an object tracked in one image to the world coordinate system by using the terrain map [vi], but such approach would fail if, for example, a person jumped or if there are errors in locating the bottom of the object. The algorithm described in [vii] uses multiple cameras, but people are tracked only by one at the time and when an object is leaving the field of view of that camera, it is handed off to the one that currently covers the object of interest. The method described in this paper assimilates the information from multiple cameras simultaneously. It uses cameras with highly overlapping fields of view (an example with four cameras is given in Figure 4).

For every frame in the video sequence, a set of centroid locations for segmented objects in each of the camera images is available. We use a segmentation algorithm by Sudderth et al. [viii] that runs at about 7 frames per second on a Pentium II-400 PC. It is a background subtraction method that adapts to the light changes by continuously updating the background pixel statistics. Foreground pixels are detecting by hypothesis testing. Segmentation detects as foreground both new objects that occlude background pixels as well as holes left by removed objects that used to belong to the background. Stationary objects and holes disappear after a certain number of frames as they are absorbed into the background. The tracker tracks both new objects and holes left by removed objects since segmentation does not differentiate between the two. Activity recognition part of the system that performs analysis of the tracks and of raw video reasons about events that produced such measurements.

Figure 4. Typical views from four cameras

Camera calibration parameters are also available. The tracker maintains a list of Kalman filters, one for each object in the scene. Since our algorithm was designed as part of a system for video surveillance, one of the requirements for the tracker was to produce updated and predicted positions of each object for the *current* frame. The availability of up-to-date prediction allows us to feed back the information to the segmentation algorithm, which can increase its sensitivity in the areas where objects are expected to be present. Also, up-to-date tracking is required to reduce the overall delay in event detection, a desirable feature in surveillance.

Several problems need to be solved for successful 3D tracking. One is to match measurements from different cameras that originate from the same object. We also need to solve the data association problem, i.e. the association of new measurements to established tracks. The number of objects is not known or constant, so the tracker also has to handle problems of track initialization and termination. In the following sections, each of these problems will be addressed.

The tracking algorithm

Camera calibration is performed once, when the cameras are placed in the room. Tsai method [ix] is used. Placing a grid of known dimensions in the room and taking the images with all four cameras, enables estimation of the parameters in the Tsai's model. The grid defines the world coordinate system, with xy plane being the grid plane (usually placed on the floor) and z coordinate being the height.

The equations that relate computer image coordinates (X, Y) to world coordinates (x_W, y_W, z_W) can be easily derived from [ix]:

$$aX + bX^3 + cXY^2 = \frac{r_1 x_W + r_2 y_W + r_3 z_W + T_x}{r_7 x_W + r_8 y_W + r_9 z_W + T_z} = o_1$$

$$dY + eY^3 + fX^2Y = \frac{r_4 x_W + r_5 y_W + r_6 z_W + T_y}{r_7 x_W + r_8 y_W + r_9 z_W + T_z} = o_2$$

(1)

where r's and T's are entries to the rotation matrix and translation vector respectively and a, b, c, d, e and f depend on other calibration parameters which include radial lens distortion. Obviously, the relationship between world and image coordinates is nonlinear.

Matching measurements from different cameras that originate from the same object

For known image coordinates of the object, to solve for the world coordinates, one has two equations with three unknowns:

$$(r_7 o_1 - r_1) x_W + (r_8 o_1 - r_2) y_W + (r_9 o_1 - r_3) z_W = T_x - T_z o_1$$

$$(r_7 o_2 - r_4) x_W + (r_8 o_2 - r_5) y_W + (r_9 o_2 - r_6) z_W = T_y - T_z o_2$$

(2)

$$\underset{2 \leftrightarrow 3}{\mathbf{A}} \underset{3 \leftrightarrow}{\mathbf{x}_w} = \underset{2 \leftrightarrow}{\mathbf{b}}$$

A point in each camera provides two equations. If an object is observed in two or more cameras, the number of equations becomes greater than the number of unknowns, and we can solve for the world coordinates using least squares:

$$\underset{2 noCameras \leftrightarrow 3}{\mathbf{A}} \underset{3 \leftrightarrow}{\mathbf{x}_w} = \underset{2 noCameras \leftrightarrow}{\mathbf{b}} \quad \mathbf{x}_w \cup (\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T \mathbf{b}$$

(3)

The residual

$$R_x = \| \mathbf{b} - \mathbf{A} \mathbf{x}_W \|_2$$

(4)

should be very small (ideally zero) if points observed in different cameras belong to the same location in the 3D world. However, for two or more points that are not projections of the same 3D point, the residual should be significant. This fact is used by the tracker which thresholds the residual to decide if the object centroids in different cameras belong to the same object in the world. It is, of course, possible to have a small residual for the points that do not belong to the

same 3D location (analogous to the case of two cameras when two objects lie in the same epipolar plane). This, however, is very unlikely to pose a problem for the tracker since predictions from already established tracks resolve such cases. Multiple objects would have to enter the scene at the very same frame and at special locations for a problem to occur and even then a potentially false track would quickly die off as the two objects start moving. In practice, we have not seen this simple approach fail.

Though the transformation from image to world coordinates is nonlinear, there is no need to use the extended Kalman filter. An intermediate result in the process of matching centroids from different cameras are the three-dimensional world coordinates of the object. These make an obvious choice as the measurements of the Kalman filter, since motion can be conveniently described in the world coordinates. This makes the design of the Kalman filter very straightforward.

Instead of immediately trying to match all combinations of centroids from different cameras, we first look for those close to predictions from the existing tracks, reducing the number of combinations that need to be examined. For each track, the measurement prediction is projected onto all four image planes and the closest centroid to each of the projections is identified. These are then used to calculate the world coordinates of the object. If the residual is bigger than the threshold, sub-residuals that correspond to pairs of equations that come from each camera are calculated and the centroid that corresponds to the largest one is discarded. Then, the world coordinate calculation is again performed with the remaining centroids and the procedure is repeated until the residual falls below the predetermined threshold or until only one centroid is left.

If only one centroid is left, we have an underdetermined system and then choose the most likely solution, the one that minimizes the Mahalanobis distance from the measurement prediction. We assume that measurement z has normal distribution with mean $\hat{z}$ and covariance matrix S, therefore we minimize the Mahalanobis distance $d^M = (z - \hat{z})^T S^{-1} (z - \hat{z})$, subject to $Az=b$. If the resulting Mahalanobis distance is smaller than a threshold that defines the validation region, the measurement is accepted.

The produced measurement is stored in the measurement list and the final association decision is made after all measurements are computed. Assigning measurements to tracks that helped localize them at this stage would amount to a nearest neighbor filter.

When the procedure described above is completed for all existing tracks, the number of centroids left to be matched is greatly reduced. Now, all possible combinations of those centroids are examined in search of possible new objects. All measurements obtained in this search are also stored in the measurement list. The residual associated with each measurement and the number of cameras from which it was computed are used to estimate the uncertainty of the measurement location.

Kalman filter design

To model dynamic behavior of objects moving in the scene, the following equations for each of the three world coordinates are used:

$$\xi[k+1] = \xi[k] + T\frac{1}{2}(\dot{\xi}[k] + \dot{\xi}[k+1])$$
$$\dot{\xi}[k+1] = \dot{\xi}[k] + T\alpha[k] \qquad (5)$$
$$E[\alpha[k]] = 0$$
$$E[\alpha^2[k]] = q$$

where _ is one of the three world coordinates and T is the sampling period. The acceleration _ is modeled as a zero-mean Gaussian random variable. The variance of the acceleration is a parameter that can be modified by the user. The system and measurement equations of the Kalman filter are:

$$\mathbf{x}[k+1] = \mathbf{F}\mathbf{x}[k] + \mathbf{v}[k] \qquad (6)$$

$$\mathbf{z}[k] = \mathbf{H}\mathbf{x}[k] + \mathbf{w}[k] \qquad (7)$$

Therefore, if we choose

$$\mathbf{x} = \begin{bmatrix} x_w & y_w & z_w & \dot{x}_w & \dot{y}_w & \dot{z}_w \end{bmatrix}^T \text{ and}$$

$$\mathbf{z} = \begin{bmatrix} x_w & y_w & z_w \end{bmatrix}^T$$

F, H and the covariance matrix for v are very easy to derive. S, the covariance matrix for w, is chosen to be $\_^2I$, where _ is the variance of the object centroid location. There is no reason to believe that errors in the three directions would be correlated or have different variances. We estimated _ by comparing measured centroids to those calculated from points marked by hand in few sequences.

Data association problem

In the presence of multiple objects, occlusions and false measurements, it is important to assign the correct measurement to each track. This is called the data association problem [x,xi].

The Mahalanobis distance from the predicted measurement is proportional to the likelihood of a measurement originating from a given track. One can define the validation volume in which there is a high probability of finding a measurement, by setting a threshold on the Mahalanobis distance. The validation volume is often used for tracking multiple objects, since it can reduce the number of measurement assignment combinations that have to be examined.

Several approaches to data association have been proposed. A nearest neighbor filter [x] assigns to the track the measurement that is closest to its predicted location. In situations when objects get close to each other, this method can easily fail and lead to loss of track. The track splitting algorithm [x,xi,xii] splits the track into hypothesis tracks for every measurement that falls into the validation region, forming the track tree. This algorithm does not prevent tracks from sharing measurements, so tracks with similar states have to be merged since they are likely to originate from the same object. Measurement assignment decisions are made by discarding track tree branches, which happens several frames after the frame in which the measurement was acquired, thus introducing a delay. In the multiple hypothesis method [x,xi,xiii,xiv], a hypothesis is one possible assignment of measurements to tracks and each creates a new branch of the hypothesis tree. The hypothesis probability is used for pruning the hypothesis tree. As with the track splitting algorithm, the association decision is made with the delay of several frames. Joint probabilistic data association filter (JPDAF) [x,xi] produces for each track an innovation that is a weighted sum of innovations from all measurements. Each weight is a posterior probability that the measurement originated from the track under consideration. A strong assumption is made that the number of tracks is fixed and known.

We have first implemented the nearest neighbor filter, which was often losing tracks when objects got close together. Track splitting algorithm introduces a delay and if forced to make a decision at a current frame, amounts to the nearest neighbor filter. The multiple hypothesis filter also introduces a delay. One could force this algorithm to choose the most likely hypothesis at a current frame, but it requires the estimates of the probabilities of track detection

and deletion which are difficult to estimate as they depend on object location and many parameters that determine the sensitivity of the segmentation algorithm. That is the reason we decided to optimize a criterion that does not require these parameters.

We chose the likelihood of the measurements-tracks assignment as the optimality criterion. It is the probability of the encountered measurements $Z(k)$ given the assignment under consideration, $\_(k)$:

$$\Lambda(\theta(k))= p(Z(k)/\theta(k))= \mathop{\subseteq}_{i=1}^{m(k)} p(z_i(k)/\theta(k))= \mathop{\subseteq}_{i=1}^{m(k)} \left\{f_{t_i}(z_i[k])\right\} V^{-(1-\tau_i)}$$

$$= V^{\tau-m(k)}c\exp - \mathop{}_{i=1}^{m(k)} \left(d_i^M\right)^{\tau_i} \,\text{?}_{\text{?}} \qquad (8)$$

where d_i^M is the Mahalanobis distance of $z_i[k]$ from the measurement prediction for track t_i, $m(k)$ is the number of measurements, $\_i$ is the indicator of whether measurement i was assigned to an existing track, $\_$ is the number of measurements that are assigned to existing tracks, $f_{ti}(z)$ is a Gaussian pdf of a new measurement for track t_i and V is the observation volume.

New objects can appear anywhere in the environment since they can be people entering the scene, objects left by people, moved objects and holes where the moved objects used to be. False measurements can come from shadows, object that are close together and are segmented as one object by segmentation, etc. It is, therefore, assumed that the probability distribution of false measurements and new tracks is uniform in the observation volume. The multiple hypothesis algorithm also requires these parameters.

Therefore in (8), $p(z_i(k)/\theta(k))= f_{t_i}(z_i[k])$, if $z_i[k]$ is assigned to an existing track t_i by $\_(k)$. If a measurement is marked by $\_(k$ as originating from a false measurement or a new track, then $p(z_i(k)/\theta(k))=V^{-1}$.

To maximize the defined optimality function, we need to minimize the sum of the Mahalanobis distances for measurements that are assigned to existing tracks and also to maximize the number of measurements that get assigned to existing tracks. The search for the optimal solution is done by constructing a *distance table*. This is a matrix where each row represents a track and each column represents a measurement. The Mahalanobis distances for all track-measurement pairs are computed and stored in this matrix. Values larger than a given threshold are discarded and replaced with a very large number (measurement outside the validation region). Then, a search through the table is performed and the combination of track-measurement pairs which minimizes the sum of Mahalanobis distances is chosen (tracks are not allowed to share measurements). If the search assigns to a track a measurement to which the distance is replaced by a large number, it simply means that there is no good measurement for that track. An example of this procedure is given in Figure 5.

Track initialization. The measurements that are not assigned to existing tracks will be used to initialize new tracks. However, even though a new Kalman filter is initialized and added to the list of filters, the track is considered established only after it was assigned a measurement for a fixed number of consecutive frames. If during those first few frames, appropriate measurements were not found, the track is discontinued and it never enters the output list of established tracks.

Track termination. Occlusion is possible in the tracking scenario described (objects that are too close together are segmented as one object). If no measurements were found for a track, it is continued for a fixed number of consecutive frames. If the measurement does not appear during that time, the track is terminated.
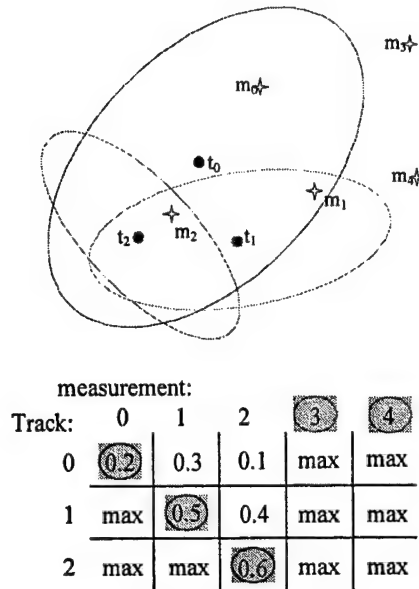
Figure 5. Example of data association. Top: t_i is the i-th track's predicted position, m_i is the i-th measurement and ellipses illustrate the validation regions. Bottom: Distance table. Circled entries in the distance table indicate optimal matches. Circled measurement number indicates that a measurement will start a new track. Note that neither of the tracks 0 and 1 get assigned measurement 2 even though it is the closest measurement to both. Nearest neighbor filter would assign it to track 0 and that would leave track 2 with no measurement.

Results

We applied the described algorithm to both synthetic and real data. Synthetic paths were created using the system equation of the Kalman filter (6) and adding sharp turns. The noise was added according to the measurement equation (7) and the measurement was then projected to the four image planes. These points were then used as input to the algorithm. Tracking was evaluated for one and two objects present in the scene. When projecting to the image plane, subpixel locations were rounded off since the segmentation algorithm used for real data also gives integer coordinates of object centroids. This can introduce error in computed world coordinates of up to 30mm (we ran simulations to estimate this error).

To produce real data, we marked three different paths on the floor of the room (Figure 6) and measured their positions with respect to the 3D coordinate system.
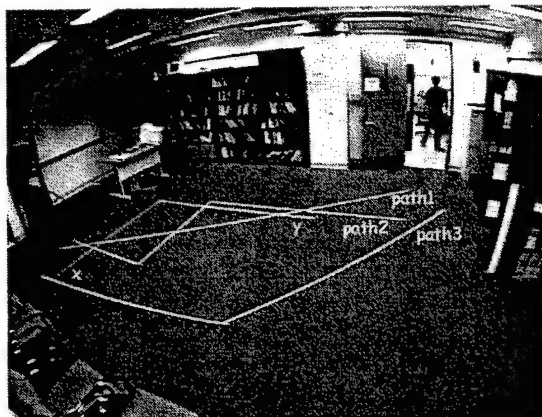
Figure 6. The world coordinate system and the measured paths

First set of sequences was recorded with three different persons walking each of the paths. For the second set, two people would walk at the same time on two different paths. Four cameras were attached to a video quad, which was connected to a Pentium II-400MHz PC. AVI movies of image size 320×240 were grabbed, thus one camera image was grabbed at 160×120. The system runs at about the same rate it performs at without the tracker (segmentation and display of object centroids over the original video). It is hard to notice any degradation in speed with the tracker running.

All sequences, both synthetic and real were tracked with the same set of parameters that were tuned on real sequences. Sampling period was 0.1s.

First three experiments were done with each of the three paths being walked by three different people. Figure 7 shows results of tracking them walking path 2. Table 1 gives mean and standard deviation of the error. Negative error means that track was closer to (0,0) than the ideal path.
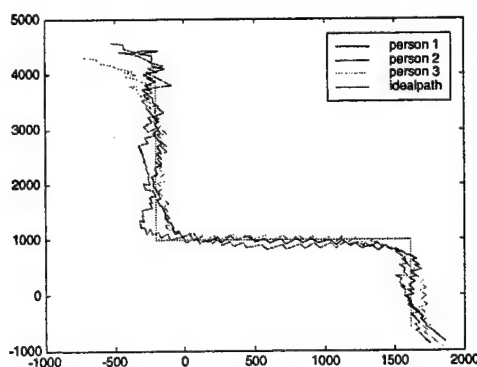

Figure 7. xy plots of the ideal and tracked paths for three persons walking path 2

Mean/ Std.dev.[mm]	Path1	Path2	Path3
Pers on1	6 .2/39.6	- 8.2/79.8	6 3.2/71.8
Pers on2	4 7.4/60.4	- 7.5/87.9	9 .6/72.4
Pers on3	3 0.4/29.2	- 36.2/114.5	3 4.2/58.9

Table 1. Mean and standard deviation of the error for tracks with one person in the scene

Figure 8 shows an example frame from one sequence with two people. First image shows the captured frame, second shows segmentation results and third shows projections of the 3D tracks back on the image planes. Segmentation produces many errors as seen in the second image. In two cameras, the two persons were close and were segmented as one object. In one camera, colors of persons' legs were similar to the background and were not segmented, therefore centroids in that camera were higher than the true centroids. Measurements in one image were accurate and the objects are successfully tracked.

(a)

(b)

12
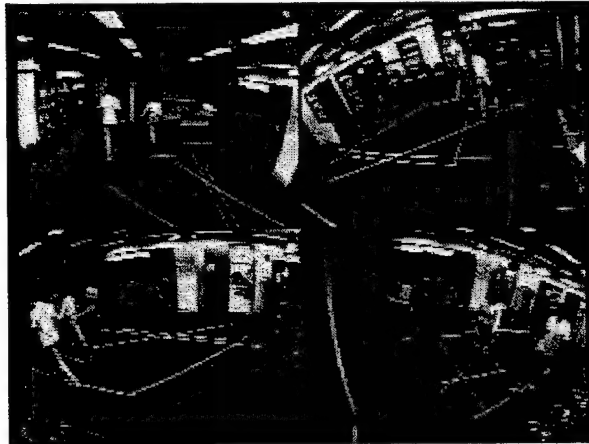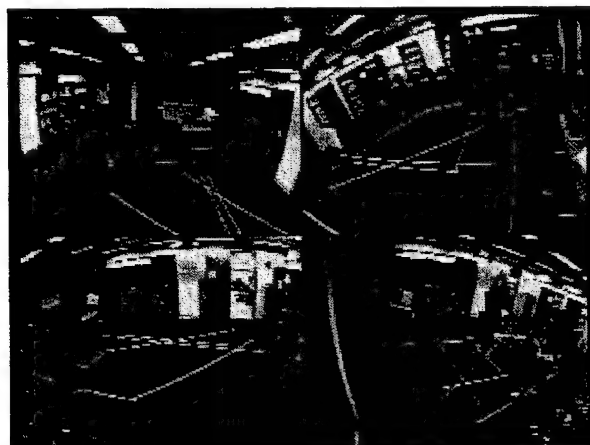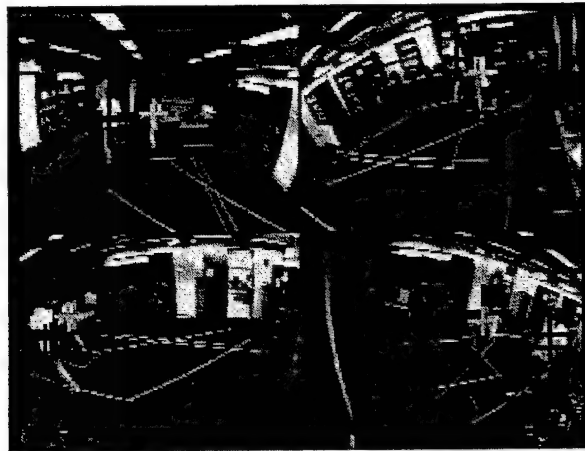
(c)

Figure 8. An example from the test sequence. (a)Captured images from four cameras. (b)Segmentation results. Red crosshairs are centroids of segmented regions larger than a threshold. (c)Tracking results. Blue and yellow crosshairs are projections of 3D tracks back onto the image planes.

Query Subsystem

The query subsystem allows user to access the data in the EM, and provides facilities to store and query historical data. The architecture of the query subsystem is shown in Figure 9.

The data from the Log-book are transformed into a state transition diagram that describes the dynamic evolution of the object with respect of the static model. The extraction of the state model is done by the integration of a series of "feature change" detectors and a rule system. The feature change detectors are application specific, and are attached to the database by the application developers. They keep track of the object features represented in the log book, and at any time give the rule system a measure of the confidence that a significant change took place in the feature they track.

The detectors keep track of the evolution of certain features of the objects (e.g., their shape, their velocity, or their color) and alert the rule system when a significant change occurs. The rule system determines whether the change is significant to determine a change of state. At the same time, the rule system keeps track of the position of the object, and may determine a change of state when an object enters or leaves one of the regions of interest.
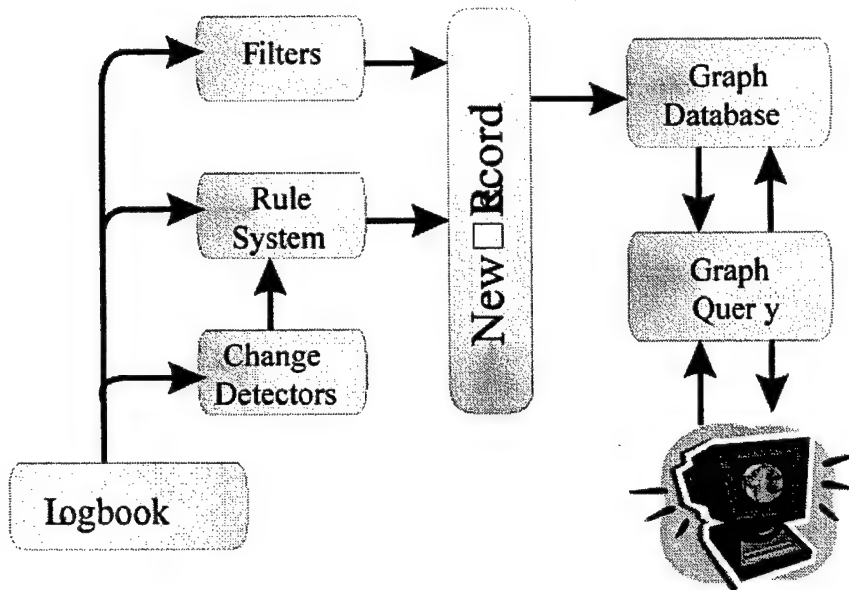
Figure 9

In the instance of Figure 10, the region of interests are A and B.
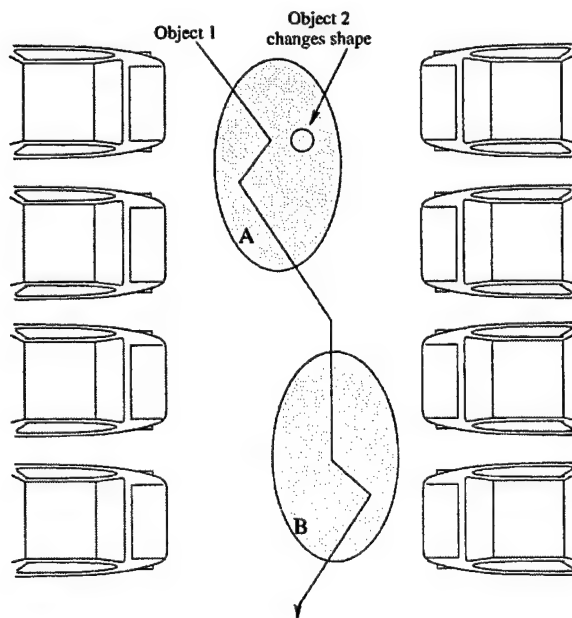

Figure 10

Object 1 starts in state "moving in region A." When the object leaves region A its state changes to "out of regions." Then, when it enters region B, its state changes to "moving in region B", and so on for the following changes of regions. The sequence of regions visited can be used

14

to identify certain behavior. The behavior of Figure 10 is definitely suspicious: the pattern of region changes agrees with that of a person looking for an open car to steal.

Object 2 (a pedestrian) is standing in region A. When he lies down, the change in shape is detected by a shape tracking algorithm, and the algorithm triggers a rule changing the state to "lying outside the regions."

When a state change is detected, the rule systems triggers the creation of a new record. The record is formed attaching the new state, the position of the object, and the time stamp to the record of the Log-book. The record in the Log-book can be quite large, since it contains visual features extracted by the video processing layers. In order to arrive to a more efficient representation, the user is allow to insert *filters* between the Log-book and the storage process. Filters can be used to synthesize information that doesn't need to be stored in full. For instance, the Log-book might contain a color histogram of the objects in the scene. On the other hand, the application designer has decided that all the database needs is information about whether an object is "reddish," "bluish," or none of the above.

The designer can then create a filter that takes the color histogram field of the Log-book record, processes it, and extracts the required information which can be stored as a simple label. This applies only to ancillary features attached to the objects. Space and time are treated differently. The position of the center of every object and the time stamps go directly to the database. This happens because---in a somewhat Kantian perspective---we consider space and time as general a priori concept that unify the whole operation of video databases, and not simple features that are accidentally extracted from the objects.

The Temporal Database

Let us assume that the event database consists of a set of tuples where each tuple has attributes (TimeStamp, ObjectId, RegionId, Event). The "TimeStamp" attribute has the domain of all real numbers as its domain. In this simplified example, "Event" attribute has the set {"Enter", "Exit", "StartLeft", "EndLeft", "StartRight", "EndLeft", "LoudNoise", "Stop", "ExitCar"} as its domain. These events relate to the movement of the centroids of objects in the environment, or other sensory data related to objects. Table 1.1 shows a snapshot of the database with 10 tuples. We use the numbers in the first column to identify the tuples in the discussion that follows, they are not part of the database instance.

	TimeStamp	ObjectId	RegionId	Event
1	1.5	1	5	Enter
2	1.2	4	2	Exit
3	2	1	5	Startleft
4	5	2	3	Exit
5	3.2	3	5	StartLeft
6	4.3	2	3	Enter
7	3	1	5	EndLeft
8	1.5	2	2	EndRight
9	5.5	1	5	StartRight
10	7	1	5	EndRight

Table 1.1: A simplified snapshot of an event database in an activity recognition system.

Like most event models, we would like to capture the fact that tuples in this database do not form a set, because they are temporally ordered. However, we may represent this temporal order under different ordering assumptions. Let us construct a temporal relation called *occurs_before* and place the tuples in the relation.

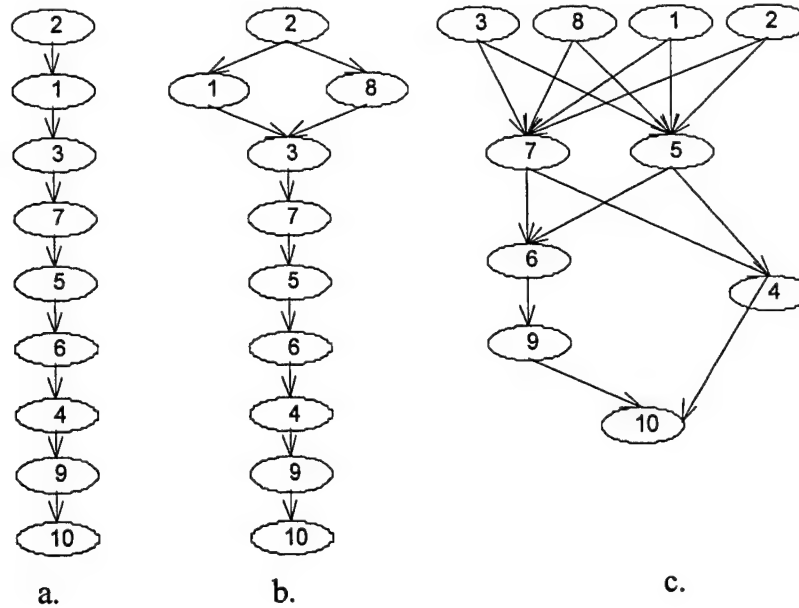**Total Order**: The first and most naïve representation is that of total order (TO) shown in

Figure 1 1: Total order, weak order, and semiorder
representations of the environment

Figure 1 1(a). Here *x occurs_before y*, if and only if timestamp of *y* is *strictly greater* that that of *x*. A representation of the environment instance as a totally ordered set is shown in figure 1 1(a). In this figure, the node numbers correspond to the tuple numbers in table 1.1. Note that our environment instance is not completely modeled by a totally ordered set since both tuple 1 and tuple 8 cannot be present at the same time.

Weak Order: The problem above can be rectified by including the notion of concurrent events. Two events are said to occur concurrently if they have *exactly* the same timestamps. Figure 1 1(b) depicts the representation of our example environment based on this "point-based" definition of concurrency and "occurs_before" relation. Such ordered sets are known as weakly ordered (WO) sets. In this representation, although we recognize concurrent events, we impose the condition that concurrency is *transitive*, i.e., if *A* is concurrent to *B* and *B* is concurrent to *C* then *A* and *C* are concurrent. As discussed in Section 2, this is the most common way of modeling simultaneous events in temporal database models.

Semiorder: The transitivity assumption of weakly ordered sets does not adequately model many applications. In activity recognition systems, the timestamps on events are always approximate, and depend on parameters like the hysteresis built into the algorithms executed by signal processing modules. So we relax the definition of concurrency by assuming that there is an interval Δ of uncertainty associated with an event occurrence, and this interval is fixed across a given set of events. Two events are concurrent if they occur within Δ of each other. This assumption gives rise to an ordered structure over the set of tuples where concurrency of events is not transitive, and is known as semiorder (SO) set. A semiorder representation of the example is depicted in figure 1 1 (c) under $\Delta=1$, and tuples 4, 6 and 9 show the effect of removing the transitivity assumption.

The event model described in this paper considers the universe of events to be structured as a semiorder set. Of course, more complex temporal ordering structures can be obtained by relaxing the assumption that the interval of uncertainty is fixed across the given set of events. If this interval could take any value, then the corresponding order is known as interval order (IO). In section 2 we show that most of the event composition models in literature are either operational systems representing weak order, or very expressive systems with high computational complexity.

16

Example Queries: Next, let us present a set of increasingly complex queries for the system. Queries 4, 5, 6, and 7 refer to figure 1.2 which shows an example of a complex activity
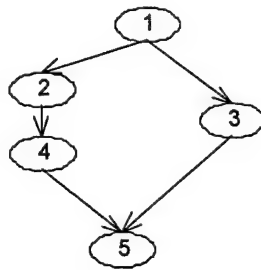

Figure 1.2: Pattern for various query examples

query in the multi-sensory monitoring environment. The query represents the activity of a car "hitting" a wall. Here nodes may represent a car entering the environment (node 1), car entering the region near the wall (node 2), car stopping (node 4), occurrence of loud noise in the region near the wall (node 3), and finally a person coming out of the car (node 5). In these queries, we further simplify by not specifying the region attribute etc. which represent the environment, region near wall etc.

1. (CONCURRENT) Find all events that occur concurrently with object 2 exiting region 3.

2. (LAST) Find the last event associated with object 2.

3. (NEXT) Find the "immediate" next event(s) of object 2 after it exited region 3.

4. (UNLABELED PATTERN) Find an activity whose temporal structure is depicted by the semiorder in figure 1.2.

5. (LABELED PATTERN) Find the activity, whose temporal structure is depicted in figure 1.2, and its event node labels are specified as below:

> node 1 label : (Event = "Enter"); //comment: Enter the environment
>
> node 2 label: (Event = "Enter"); //comment: Enter the region near the wall
>
> node 3 label : (Event = "LoudNoise");
>
> node 4 label: (Event = "Stop");//comment: car stops
>
> node 5 label: (Event = "ExitCar");

6. (LABELED PATTERN with Local Constraint) Find the activity whose temporal structure is depicted by the semiorder in figure 1.2, and its set of event labels are:

> node 1 label : (Event $\in$ {"Enter", "StartLeft"});
>
> node 2 label: (Event $\notin$ {"Exit"});
>
> node 3 label : (Event = "LoudNoise");
>
> node 4 label: (Event = "Stop");
>
> node 5 label: (Event = "ExitCar");

7. (LABELED PATTERN with Global Constraint) Find the activity whose temporal structure is depicted by the semiorder in figure 1.2, and its set of event labels are:

> node 1 label : (Event = "Enter");
>
> node 2 label: (Event $\notin$ {"ExitRegion"});

node 3 label : (Event = "LoudNoise");

node 4 label: (Event = "Stop");

node 5 label: (Event = "ExitCar");

with the global constraint given by the predicate p: (Node4.RegionID = Node3.RegionID)

8. (MAXIMAL) Find the maximal common activities of object 1 and object 2 between times T1 and T2.

This set of queries is ordered according to the increasing computational difficulty of answering them in the semiorder data model.

These queries illustrate the need for a number of basic properties in the query language we develop:

- it answers all relational queries.

- recursion is supported for temporal attributes to express queries, like CONCURRENT and NEXT, under the assumption of uncertainty intervals.

- the language includes a "pattern description" component where a pattern is a set of semiorders.

- it support aggregate operators.

Results

We have been working on two scenarios: the surveillance of a room and the surveillance of a parking lot. For the first scenario, the system has been installed in a library adjacent the laboratory. Four cameras are used to monitor the activity in the room. We defined several regions of interest and placed objects to be monitored in the room. The system works on two PCs and monitors activities like "stealing an object," "loitering near a bookshelf," "jumping on a table."

A second system was installed on a parking lot outside the lab, and includes three cameras. It monitors activities like pedestrians going near several cars, cars entering the parking lot at excessive speed, and cars remaining parked in the same spot for more than a specified amount of time.

References

[i] C. Wren, A. Azarbayejani, T. Darell, A. Pentland. Pfinder: Real-Time Tracking of the Human Body. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 19(7):780-785 July 1997

[ii] I. Haritaoglu, D. Harwood, L. S. Davis. W[4]: Who? When? Where? What? A real time system for detecting and tracking people. Proc. Third IEEE Int. Conf. on Automatic Face and Gesture, Nara, Japan, April 1998, p.227-227

[iii] C. Bregler, J. Malik. Tracking People with Twists and Exponential Maps, *CVPR98, p.*8-15

[iv] D. M. Gavrila and L.S. Davis. 3-D Model-based Tracking of Humans in Action: a Multi-view Approach. *CVPR96*, p.73-80

[v] T. Horprasert, I. Haritaoglu, D. Harwood, L. Davis, C. Wren, A. Pentlnad, "Real-time 3D Motion Capture", PUI98, November 1998

[vi] T. Kanade, R. T. Collins, A. J. Lipton. Advances in Cooperative Multi-Sensor Video Surveillance, *Proc. DARPA Image Understanding Workshop '98*, p.3-24

[vii] Q. Cai and J. K. Aggarwal. Tracking Human Motion using Multiple Cameras, *Proc. Int. Conf. on Pattern Recognition 1996*, p.68-72

[viii] E. Sudderth, E. Hunter, K. Kreutz-Delgado, P. Kelly, R. Jain, "Adaptive Video Segmentation: Theory and Real-Time Implementation", *Proc. DARPA Image Understanding Workshop '98*, p. 177-181

[ix] R. Tsai. A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses. *IEEE Journal of Robotics and Automation, vol. RA-3, no 4, August 1987*, p. 323-344

[x] I. J. Cox. A Review of Statistical Data Association Techniques for Motion Correspondence. Int. Journal of Computer Vision 10:1 1993, p.53-56

[xi] J. Bar-Shalom, T. E. Fortmann. Tracking and Data Association, Academic Press, 1988

[xii] Z. Zhang, O. D. Faugeras. Three-Dimensional Motion Computation and Object Segmentation in a Long Sequence of Stereo Frames. *Int. Journal of Computer Vision, 7:3, 1992* p. 211-241

[xiii] D. B. Reid. An Algorithm for Tracking Multiple Targets, *IEEE Trans. Automatic Control, vol. AC-24, no. 6, December 1979*, p. 843-854

[xiv] I. J. Cox, S. L. Hingorani. An Efficient Implementation of Reid's Multiple Hypothesis Tracking Algorithm and Its Evaluation for the Purpose of Visual Tracking. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 18(2):138-150, February 1996

AFRL/IFED (PETER J. COSTIANES)
32 BROOKS ROAD
ROME NY 13441-4114

10

AFRL/IFOIL
26 ELECTRONIC PARKWAY
ROME NY 13441-4514

1

ATTN DTIC-OCC
DEFENSE TECHNICAL INFO CENTER
8725 JOHN J. KINGMAN ROAD, STE 0944
FT BELVOIR VA 22060-6218

2

ATTN DR RAMESH JAIN
DEPT. OF ELECTRICAL AND COMPUTER ENG.
UNIVERSITY OF CALIFORNIA, SAN DIEGO
LA JOLLA CA 92093-0934

5

ATTN GEORGE LUKES
DEFENSE ADVANCED RESEARCH PROJECTS AGENCY
DARPA/ISO
3701 N. FAIRFAX DRIVE
ARLINGTON VA 22203-1714

5

MISSION
OF
AFRL/INFORMATION DIRECTORATE (IF)

The advancement and application of information systems **science and** technology for aerospace command and control and its **transition to air,** space, and ground systems to meet customer needs in the **areas of Global** Awareness, Dynamic Planning and Execution, and Global **Information** Exchange is the focus of this AFRL organization. The directorate's areas of investigation include a broad spectrum of information and fusion, communication, collaborative environment and modeling and simulation, defensive information warfare, and intelligent information systems technologies.